электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.418

Интеграция корпоративных информационных систем с помощью асинхронного обмена сообщениями

Селезнев А.С., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Системы обработки информации и управления»

Научный руководитель: М.В. Виноградова, к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
vinogradova@bmstu.ru

Сегодня интеграция приложений и бизнес-процессов является приоритетной задачей для многих предприятий. Требования к улучшению качества услуг и быстро меняющаяся бизнес-среда являются основными причинами для расширения взаимодействия между существующими системами. Очень мало новых бизнес-приложений в настоящее время разрабатываются или развернуты без основательного подхода к интеграции, что делает её определяющим качество свойством для приложений.

Существует много различных инструментов и технологий, которые доступны для решения неизбежных сложностей интеграции разрозненных систем. Новейшие технологии, спецификации и стандарты веб-сервисов усилили акцент на интеграции технологий.

Разработка архитектуры интеграционного решения является сложной задачей. Нельзя точно сказать был ли выбор архитектуры верный пока не пройдут месяцы или даже годы. На предприятиях, как правило, используется множество приложений, которые могут быть разработаны внутри компании, приобретены у третьих сторон, быть частью старой системы, работать в различных операционных системах.

Распределение функционала между несколькими приложениями дает предприятию изрядное количество гибкости, поэтому очень важно правильно подобрать нужные компоненты для различных целей, таких как система заказов, система управления взаимоотношениями с клиентами и т.п. В целях поддержки общих бизнес-процессов и обмена данными между приложениями, используются различные методы, такие как обмен файлами, обмен информацией на уровне базы данных, удаленный вызов и асинхронный обмен сообщениями.

Обмен файлами достаточно распространенный подход к организации взаимодействия. Это связано с относительной простотой реализации, а также существованием стандартных форматов обмена данными. Например, большая часть корпоративных информационных систем позволяет загружать и выгружать файлы, в формате CSV (Comma-Separated Values – «поля, разделенные запятыми»). У этого подхода есть множество недостатков. Так, если необходимо оперировать сложными структурами, то простые форматы обмена уже не пригодны. Кроме того, обычно обмен файлами подразумевает такие этапы как выгрузку файла из одной системы и загрузки в другую, что усложняет полноценную автоматизацию процесса и не позволяет осуществить обмен данными в реальном времени.

Обмен информацией на уровне базы данных это процесс, который включает в себя извлечение данных из одной системы и загрузки в другую. При необходимости данные приводятся к нужному формату для каждой системы. Некоторые СУБД (Microsft SQL Server, Oracle SQL) обладают встроенными средствами для реализации этого подхода. Этот способ позволяет реализовать обмен данными по расписанию, но не в реальном времени.

Удалённый вызов процедур позволяет программному коду, который выполняется на одном компьютере, вызывать код на другом. Подход заключается в следующем: если приложению А что-то нужно от приложения Б, то приложение А вызывает функцию приложения Б. Основной недостаток удаленного вызова – требование работоспособности всех задействованных приложений в момент взаимодействия. Если возникнет ситуация, когда какое-либо звено в системе не работает – это может нарушить целостность данных. К достоинствам этого метода следует отнести выполнение всех задач в реальном времени.

В качестве используемого подхода интеграции, был выбран асинхронный обмен сообщениями. Такой способ лишён многих недостатков других традиционных методов. Так например, инициирующему передачу данных приложению не требуется ждать подтверждения, оно продолжает работать в привычном режиме, кроме того как и удалённый вызов процедур, метод позволяет выполнять задачи по обмену информации в реальном времени.

Внешнее приложение (Инициатор) отправляет сообщения на удалённое приложение. При этом, поскольку передача сообщений ведется в асинхронном режиме, не дожидаясь ответа на первое сообщение, может быть отправлено второе. В данном случае возникает несколько проблем: сообщения могут не дойти из-за каких-то технических неполадок, последовательность сообщений может сбиться, процесс на удалённом сервере

может по каким-то причинам не выполнится или завершиться с ошибкой. Для того чтобы избежать возникновения перечисленных сложностей предлагается следующее решение. На рис. 1 приведена диаграмма последовательности описанного процесса.

На приложении, которое инициирует передачу данных (Потребитель), формируется очередь сообщений готовых к отправке, где каждого из них генерируется GUID — статистически уникальный 128-битный идентификатор. Отправленные сообщения попадают в следующую очередь. Таким образом, всегда будет известен точный порядок следования сообщений и их идентификатор. Условная схема работы алгоритма представлена на рис. 2.

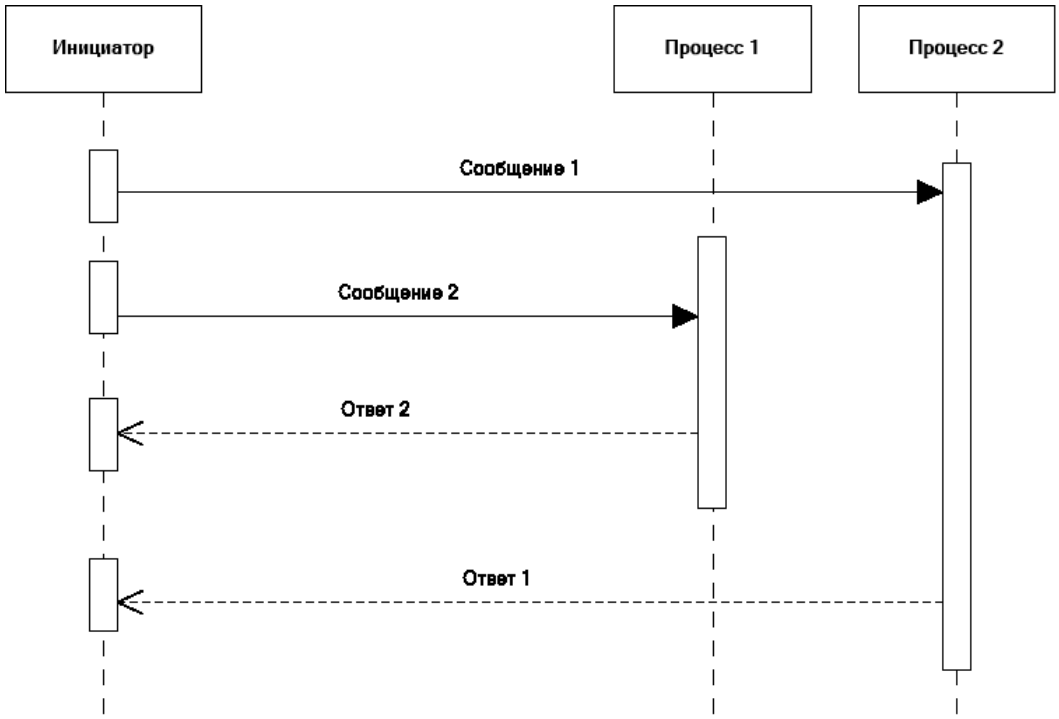

Рис. 1. Диаграмма последовательности

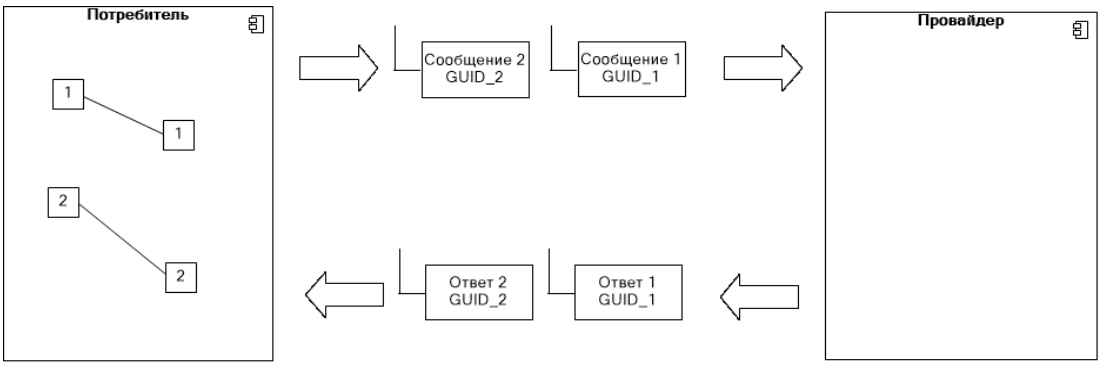

Рис. 2. Обнаружение связанных сообщений

Когда сообщение от какого-либо приложения попадает на принимающую сторону (Провайдер), запускается соответствующий процесс, например, работа с данными или проведение расчётов и т.п. После того, как процесс завершён, формируется и отправляется ответ инициирующей стороне, в который включается GUID первоначального сообщения. После получения ответа в приложении-потребителе осуществляет поиск отправленного сообщения с GUID полученного ответа. Далее такое сообщение помечается как успешно доставленное и может быть удалено, либо хранится для логирования событий. Также при необходимости выполняются определённые процессы. Например, если изменение данных должно быть произведено на обеих сторонах, то само изменение на инициирующей стороне осуществляется только тогда, когда получен ответ об успешном выполнении запроса, для которого найден соответствующий GUID.

Так как использование SOAP для передачи сообщений увеличивает их объём и снижает скорость обработки, в предлагаемой системе они передаются по протоколу HTTP с помощью метода POST, в формате XML, что позволяет добиться большей универсальности подхода и его увеличить производительность.

Такой подход позволяет осуществить асинхронный обмен данными при этом вероятность конфликтов и нарушения целостности данных сводится к минимуму. Отслеживание отправлений с помощью GUID позволят добиться безошибочной работы с базами данных, так как если на удалённом приложении произошла ошибка, то данные не будут изменены и на инициирующем сервере, потому что ответ не будет получен, или вернётся ответ с ошибкой. При этом хранение сообщений до получения соответствующего ответа об успешном выполнении процесса, позволяет восстановить работу после непредвиденных сбоев не выполняя при этом дополнительных манипуляций. Асинхронность процесса позволяет добиться более отзывчивого интерфейса и увеличить производительность системы в целом.

Разработанный алгоритм позволяет осуществить надёжную интеграцию приложений при этом достаточно универсален, и может быть доработан для объединения нескольких приложений в одну систему. Такой способ осуществления взаимодействия между различными процессами подойдёт как крупному, так и небольшому предприятию. При этом важно помнить, что асинхронные системы требуют иного подхода к проектированию архитектуры, по сравнению с синхронными.

Список литературы

1. G. Hohpe, B. Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, 2003. p. 683.
2. R. Miles, K. Hamilton. Learning UML 2.0. O'Reilly Media Inc., 2008. p. 290.
3. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). WC3, 2007. // URL http://www.w3.org/TR/soap12-part1 (дата обращения: 20.03.2013).
4. S. Chatterjee. Messaging Patterns in Service-Oriented Architecture. Cap Gemini Ernst & Young, 2004. // URL http://msdn.microsoft.com/en-us/library/aa480027.aspx (дата обращения: 20.03.2013).
5. E. Thomas. Service Oriented Architecture: Concepts, Technology, and Design. Indiana: Pearson Education, 2003. p. 171.
6. С. Мюнх, С. Фернстайн. Работа с XML-сообщениями, 2001. // URL http://citforum.ru/database/articles/oraclexmlmess.shtml (дата обращения: 20.03.2013).
7. А. Добровольский. Интеграция приложений: методы взаимодействия, топология, инструменты, 2006. // URL http://www.osp.ru/os/2006/09/3776464 (дата обращения: 20.03.2013).